Krüptoräsid (Hash- funktsioonid) ja autentimine. Kasutatavaimad algoritmid. MD5, SHA-1, SHA-2.

Erika Matsak, PhD

Nõudmised krüptoräsidele (Hash-funktsionidele)

- Krüptoräsiks nimetatakse ühesuunaline funktsioon
- Krüptograafiline sõnumilühend (Hash-kood) saadakse funktsiooni *H* abil: *h=H(M)*, kus *M* on suvalise pikkusega avatekst ning *h* on fikseeritud pikkusega kood.
- Selleks, et selle funktsiooni abil oleks võimalik teostada autentimist, peab funktsioon vastama järgmistele nõuetele:
  1) *H* peab olema rakendatav suvalise pikkusega tekstile
  2) *H* peab tulemuseks andma fikseeritud pikkusega koodi *h*.
  3) *H(M)* on suhteliselt lihtne arvutada polünomiaalse ajaga iga *M* puhul
  4) Iga koodi *h* jaoks on arvutuslikult võimatu leida *M*, sellise, et *H(M)=h*
  5) Kui on antud sisend *x,* siis selle jaoks on arvutuslikult võimatu leida $y \neq x$ selline, et *H(x)=H(y)*.
  6) Arvutuslikult on võimatu tuvastada paari sisenditest *(x,y)* lähtudes, et nende hash-koodid oleks võrdsed, ehk *H(x)=H(y)*

Funktsioon mis rahuldab tingumusi 1-5 on lihtne (tavaline) Hash-funktsioon, kui funktsioon rahuldab lisaks ka tingimust 6, siis Hash-funktsioon on tugev

Lihtsad krüptoräsid (Hash-funktsionid)

- Sisendit (tekst, fail) vaadeldakse kui **n-bitiste** plokkide jada
- Üks kõige lihtsamatest Hash-funktsioonidest on esitatav järgmise valemiga (plokkide XOR bittide tasemel)

  $C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{ik,}$ kus

  C_i - i-kohal bitt Hash-koodis, $1 \leq i \leq n$.

  k - n-bitiste plokkide arv

  b_{ij} - i-kohal bitt j-kohal plokkis

Tulemuseks on **n-pikkune Hash-kood (engl: hash-code, digest)**

Tavaliselt kontrollitakse selle meetodi abil andmete terviklust.

Selleks, et antud meetod peidaks teksti omadusi, kasutatakse ka bittide nihutamist:

Iga plokki sees teostataks nihe ühe biti võrra vasakule (<<)

Seejärel teostatakse XOR

Näide

T	01010100		00100000
a	01100001	Y	01011001
l	01101100	l	01101100
l	01101100	i	01101001
i	01101001	k	01101011
n	01101110	o	01101111
n	01101110	o	01101111
a	01100001	l	01101100

$C_i = b_{i1} \oplus b_{i2} \oplus b_{i3} \oplus b_{i4}$

TCP/IP kasutab 32bitist või 16 bitist plokki suurust

“Sünnipäeva paradoks”

- Kui on tegu inimeste grupiga, mis koosneb 23 inimesest, siis tõenäosus, et kas või kahel inimesel on sünnipäev samal päeval on suurem kui 0,5. Kui on tegu grupiga ≥60, siis selline tõenäosus on 99%, kui aga ≥365, siis 100%

Tõenäosus, et sünnipäevad on erinevad:

$$\bar{p}(n) = 1 \cdot \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{n-1}{365}\right) = \frac{365 \cdot 364 \cdots (365 - n + 1)}{365^n} = \frac{365!}{365^n (365 - n)!},$$

Tõenäosus, et sünnipäevad on samad:

$$p(n) = 1 - \bar{p}(n).$$

Intuitiivsel tasemel:

Kui 23 inimesest moodustada paare, siis kombinatoorika valemi järgi on selliseid paare 253

$$\binom{n}{k} = C_n^k = \frac{n!}{k!\,(n-k)!}.$$

“Sünnipäeva paradoks”

n	**p (n)**
10	12 %
20	41 %
30	70 %
50	97 %
100	99,99996 %
200	99,99999999999999999999999998 %
300	$(1 - 7\times10^{-73}) \times 100$ %
350	$(1 - 3\times10^{-131}) \times 100$ %
365	100 %

“Sünnipäeva paradoks”

- Olgu inimeste arv *n* ning sünnipäevade jaotus olgu ühtlane (sünnipäevade arv on piiratud ja nende esinemine on võrdse tõenäosusega)
- Kõigepealt arvutame tõenäosuse, et vaadeldavatel inimestel on sünnipäev erinevatel päevadel
- Võtame esimese inimese ning jätame meelde tema sünnipäeva. Võtame järgmise inimese. Tõenäosus, et tema sünnipäev on esimesest inimesest erinev on 1 — 1/365.

“Sünnipäeva paradoks”

- Vaatame nüüd kuidas lood on tõenäosusega, et keegi grupist on sündinud samal päeval isikuga, kes ei kuulu valitud gruppi

$$q(n) = 1 - \left(\frac{365-1}{365}\right)^n$$

- Kui *n*=23, siis selline tõenäosus on ≈5,9%. Selleks, et see tõenäosus oleks suurem kui 50%, peaks *n* olema ≥253.
- Selline erinevus on seotud asjaoluga, et grupis võivad sünnipäevad korduda ning see teeb meie arvutatud tõenäosuse väiksemaks.

“Paradoksi” kasutus Hash-funktsioonsis

- Olgu funktsiooni *H* tulemuse, ehk hash-koodi pikkus *n*.
- Milline peab olema arv *k*, selleks, et konkreetse *X* ja *Y* väärtuste *Y1, …, Yk* korral kas või ühe *Yi* jaoks oleks võimalik, et *H (X) = H (Yi)* ning selle tõenäosus oleks suurem kui 0,5.
- Kui tegu on ühe väärtusega *Y*, siis nimetatud tõenäosus, et *H (X) = H (Y),* võrdub *1/n*. Vastavalt, tõenäosus, et *H (X) ≠ H (Y)* on *1 - 1/n*.
- Kui tegu ei ole suuruse ainsa *Y* väärtusega (*Y)*, vaid *k* väärtusega *Y1,…,Yk*, siis tõenäosus, et *H (X) ≠ H (Y)* on $(1 - 1/n)^k$
- Eelnevast tuleneb, et tõenäosus, et suuruse Y mingi väärtuse korral

  H(X) = H (Y), võrdub arvuga $1 - (1 - 1/n)^k$

  $(1 - a)^k = 1 - ka + (k(k-1)/2!)a^2 - \ldots \approx 1 - ka$

  $1 - (1 - k/n) = k/n = 0,5$

  $k = n/2$

Vaatamata sellele, et kõikide võimalike Hash-koodide arv on 2^n, on võimalik meie Hash-koodile “pihta saada” ka juhul, kui Hash-funktsiooni arvutamine on läbi katsetatud $2^{n/2}$ korda.

Ründed, mis kasutavad “paradoksi” (Collision Attack)

- Vastane moodustab (mõtleb välja) $2^{m/2}$ erinevat teksti, millel on mingi konkreetne mõte ning sama palju võltstekste.
- Nendest kõikidest tekstidest arvutatakse Hash-funktsiooni abil Hash-kood ning otsitakse võrdseid paare. Tõenäosusega $>0,5$ leiame võrdse paari.
- Kui ei ole leitud, genereeritakse tekste juurde . Jätkatakse kuni vajalik paar on leitud
- Vastane annab Alice-le (saatja) allkirjastamisele õige dokumendi. Seejärel Alice allkirja lisatakse võltsdokumendile sama hash-koodiga ning saadetakse Bobile (saajale)

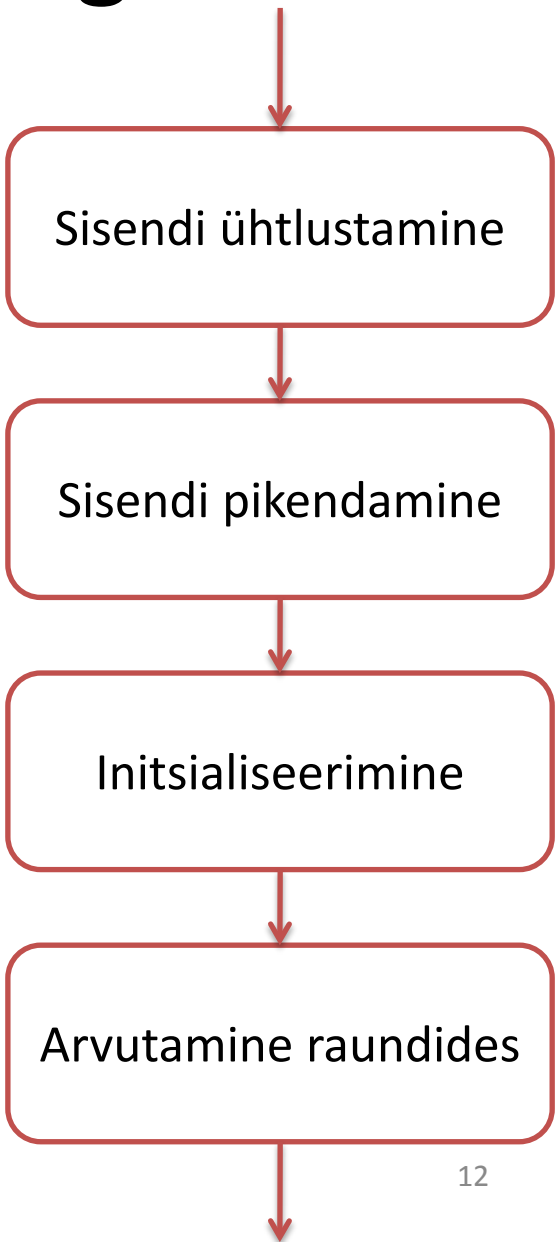
MD5 Hash-funktsiooni algoritm

- 128 bitine algoritm, töötatud välja 1991a.
- On väljaarendatud algoritmist MD4
- Teades Hash-koodi, ei ole võimalik taastada avateksti.
- 2006 on avaldatud algoritm (Vlastimil Klima) mis suudab murda MD5 ühe minutiga

Ronald L. Rivest
1947

MD5 Hash-funktsiooni algoritm

- Sisendiks on suvalise pikkusega tekst (sh nullpikkusega). Olgu sisendi pikkus *L* (mittenegatiivne täisarv).
- **Samm 1**. Sisendi ühtlustamine. Kirjutatakse sisendi lõppu bitt 1, seejärel 0-bitte. Nulle kirjutataks kuni pikkus *L'* saab võrdseks arvuga 448(mod 512). Ehk 512*N+448. Selline ühtlustamine toimib isegi siis, kui meie esialgne pikkus *L* on juba võrdne 448(mod 512) .
- **Samm2**. Sisendi pikendamine. Viimastele 64 bittide kohtadele (512-448=64) kirjutatakse arvu L (esialgne pikkus) kahendesitus. Kui $L > 2^{64} - 1$, siis kirjutatakse esimesi bitte saadust kahendkoodist.



MD5 Hash-funktsiooni algoritm. Initsialiseerimine.

- Arvutustes hakatakse kasutama korraga 4 sõna: A,B,C, D (iga sõna 32 bitti)
- Algandmed on järgmised:

  A = 01 23 45 67

  B = 89 AB CD EF

  C = FE DC BA 98

  D = 76 54 32 10

  ABCD on initsialiseerimisvektor.

  Järgmistes sammudes läheb vaja konstantide tabelit K[1..64]. $K[i] = int(4294967296 * |sin(i)|\ )$, kus $4294967296 = 2^{32}$. Teiste sõnadega: selles tabelis seisavad *sin()* funktsiooni väärtusest 32 bitti peale koma

MD5 Hash-funktsiooni algoritm. Arvutamine raundides.

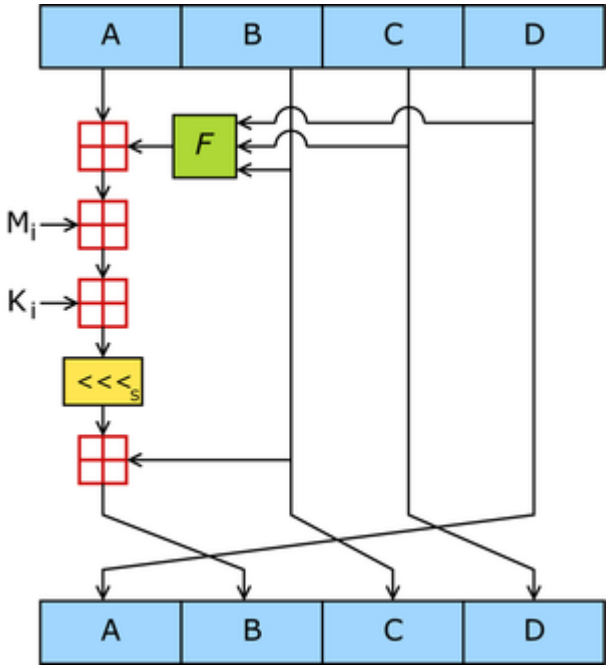
Kasutatakse 4 raundi. Igas raundis on oma funktsioon:

1 raund $F(X,Y,Z)=(X\&Y)\vee(\neg X\&Z)$

2 raund $G(X,Y,Z)=(X\&Z)\vee(\neg X\&Y)$

3 raund $H(X,Y,Z)=X\oplus Y\oplus Z$

4 raund $I(X,Y,Z)=Y\oplus(\neg Z\vee X)$



Järjekordne plokk (4 alamplokki) paigutatakse spetsialsesse massiivi (M).

Jäetakse meelde eelmise raundi ABCD (kui tegu on esimese tsükliga, siis initsialiseerimisvektor)

Kasutatakse spetsiaalset operatsiooni vastava funktsiooniga. Näiteks 1 raundi operatsioon [abcd **k s i**] tähendab

a =b + ((a + F(b,c,d) + M[**k**] + K[**i**]) <<< **s**)

Liitmine sooritatakse kasutades mod 2^{32}

MD5 Hash-funktsiooni algoritm. Arvutamine raundides. Lähemalt

- Olgu

  A = 01 23 45 67 =00000001 00100011 01000101 01100111

  B = 89 AB CD EF = 10001001 10101011 11001101 11101111

  C = FE DC BA 98 = 11111110 11011100 10111010 10011000

  D = 76 54 32 10= 01110110 01010100 00110010 00010000

[abcd **0 s 1**] a =b + ((a + F(b,c,d) + M[**0**] + K[1]) <<< **s**)

K[*1*] = *int*(4294967296 * |*sin*(*1*)|)

plokk nr i paigutatakse massiivi M: **Tallinna Ylikool, Informaatika Instituut. Narva mnt 25, Tallinn, Eesti.**

Siin k=0 on M[0]: Tall

s väärtused esimeses raundis on: 7, 12, 17, 22, ehk esimene nendest on 7

Esimeses raundis on F(B,C,D)=(B&C)$\vee$($\neg$B&D)

MD5 Hash-funktsiooni algoritm. Arvutamine raundides. Pseudokood.

```
// jagame avateksti plokkideks nii et igas plokks oleks 16 sõna:
for i = 0 to N/16 - 1 do
{
// plokk nr i paigutatakse massiivi M
for j = 0 to 15 do
        M[j] = X[i * 16 + j]
// eelmised A, B, C, D jäetakse meelde
        AA = A;   BB = B;    CC = C;    DD = D
// raund 1          /*[abcd k s i] a = b + ((a + F(b,c,d) + M[k] + K[i]) <<< s). */
        [ABCD 0 7 1][DABC 1 12 2][CDAB 2 17 3][BCDA 3 22 4]
        [ABCD 4 7 5][DABC 5 12 6][CDAB 6 17 7][BCDA 7 22 8]
        [ABCD 8 7 9][DABC 9 12 10][CDAB 10 17 11][BCDA 11 22 12]
        [ABCD 12 7 13][DABC 13 12 14][CDAB 14 17 15][BCDA 15 22 16]
// raund 2          /*[abcd k s i] a = b + ((a + G(b,c,d) + M[k] + K[i]) <<< s). */
[ABCD 1 5 17][DABC 6 9 18][CDAB 11 14 19][BCDA 0 20 20]
[ABCD 5 5 21][DABC 10 9 22][CDAB 15 14 23][BCDA 4 20 24]
[ABCD 9 5 25][DABC 14 9 26][CDAB 3 14 27][BCDA 8 20 28]
[ABCD 13 5 29][DABC 2 9 30][CDAB 7 14 31][BCDA 12 20 32]
```

MD5 Hash-funktsiooni algoritm. Arvutamine raundides. Pseudokood.

// raund 3 /*[abcd k s i] a = b + ((a + H(b,c,d) + M[k] + K[i]) <<< s). */

[ABCD 5 4 33][DABC 8 11 34][CDAB 11 16 35][BCDA 14 23 36]

[ABCD 1 4 37][DABC 4 11 38][CDAB 7 16 39][BCDA 10 23 40]

[ABCD 13 4 41][DABC 0 11 42][CDAB 3 16 43][BCDA 6 23 44]

[ABCD 9 4 45][DABC 12 11 46][CDAB 15 16 47][BCDA 2 23 48]

// raund 4 /*[abcd k s i] a = b + ((a + I(b,c,d) + M[k] + K[i]) <<< s). */

[ABCD 0 6 49][DABC 7 10 50][CDAB 14 15 51][BCDA 5 21 52]

[ABCD 12 6 53][DABC 3 10 54][CDAB 10 15 55][BCDA 1 21 56]

[ABCD 8 6 57][DABC 15 10 58][CDAB 6 15 59][BCDA 13 21 60]

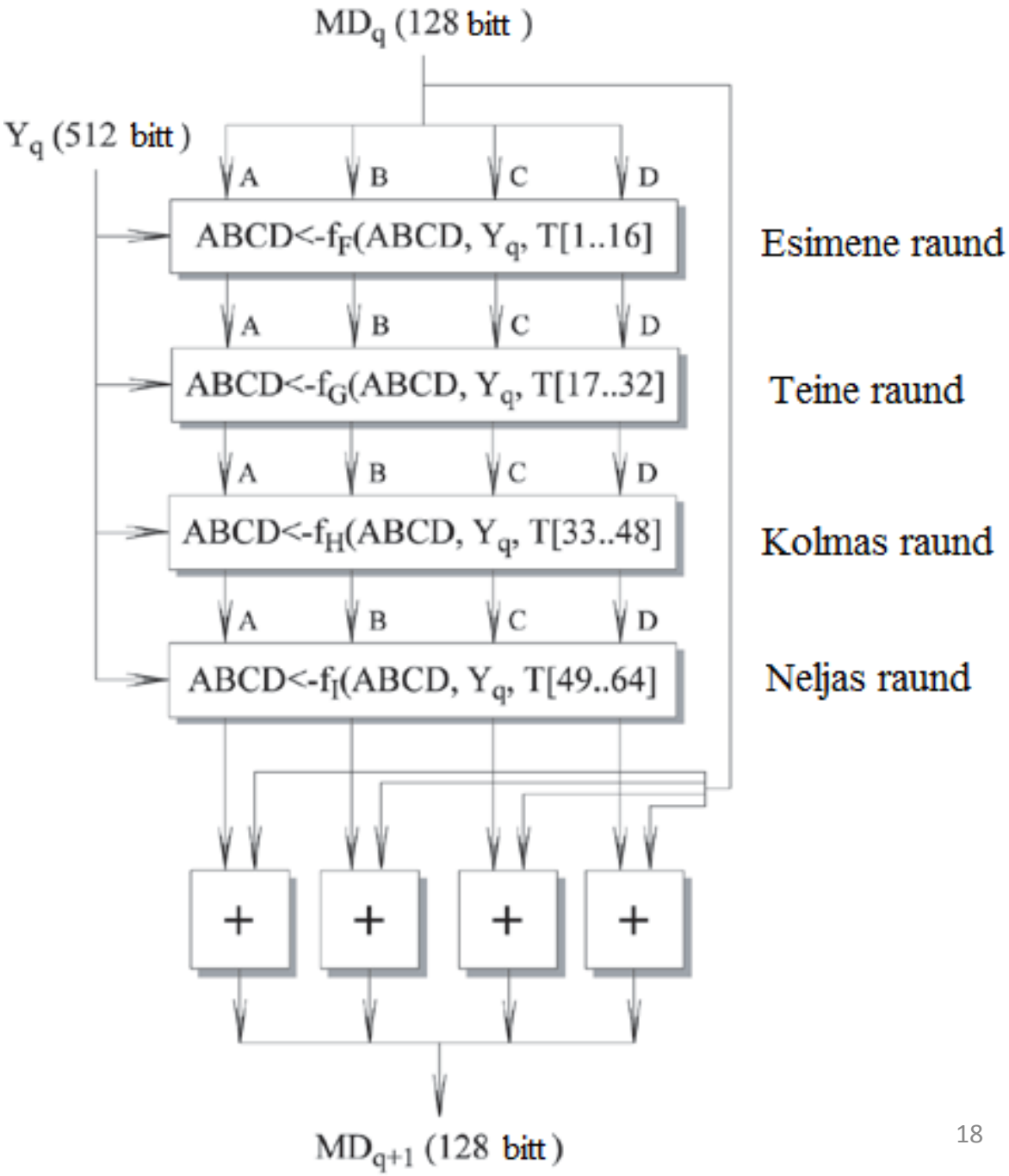
[ABCD 4 6 61][DABC 11 10 62][CDAB 2 15 63][BCDA 9 21 64]

A = AA + A; B = BB + B; C = CC + C; D = DD + D

} // for tsükli lõpp

Samm 5. Andmete väljastamine

MD5 Hash-funktsiooni algoritm. Arvutamine raundides. Plokkskeem



MD5 java

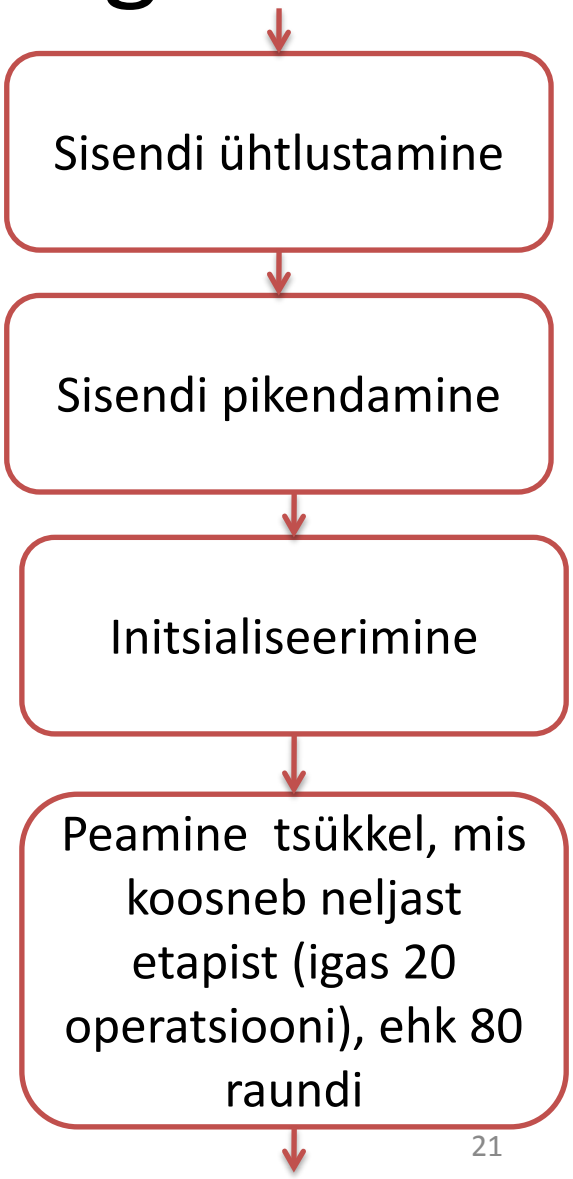
```java
import java.security.*;
import java.math.*;

public class MD5 {
   public static void main(String args[]) throws Exception{
      String s="This is a test";
      MessageDigest m=MessageDigest.getInstance("MD5");
      m.update(s.getBytes(),0,s.length());
      System.out.println("MD5: "+new BigInteger(1,m.digest()).toString(16));
   }
}
```

MD5 omadused

- Iga hash-koodi bitt on funktsioon igast sisendbitist.
- Funktsioonide FGHI kordamine kindlustab seda, et tulemus on hästi “ärasegatud”, ehk on vähetõenäone, et kaks avateksti, mis on valitud juhuslikult (isegi kui nad tunduvad sarnased), saavad ühesugused hash-koodid.
- MD5 on tugev Hash-funktsioon 128 bitise hash-koodi jaoks. Selleks, et leida kaks avateksti, millel on ühesugune hash-kood on vaja sooritada $\approx 2^{64}$ operatsiooni. Seejuures selleks, et leida avatekst etteantud hash-koodiga, on vaja $\approx 2^{128}$ operatsiooni .

SHA-1 Hash-funktsiooni algoritm.

- SHA-1 (Secure Hash Algorithm 1) on väljatöötatud 1995a.
- Antud algoritmil on palju ühist MD5 algoritmiga, sest mõlema aluseks on algoritm MD4.
- Funktsiooni sisendiks on plokk suurusega 512 bitti või eelmise tsükli tulemus
- Hash-kood, mis vastab plokile M_i on $h_i = f(M_i, h_{i-1})$. Hash koodi pikkus on 160bitti
- Terve sõnumi hash-koodiks on viimase ploki väljund.



SHA-1 Hash-funktsiooni algoritm.

- Sisendi ühtlustamine, pikendamine ja initsialiseerimine on sama, mis algoritmil MD5.
- Avatekst jagatakse plokkideks, iga ploki suurus on 512bitt.

A = 67452301

B = EFCDAB89

C = 98BADCFE

D = 10325476

E = C3D2E1F0

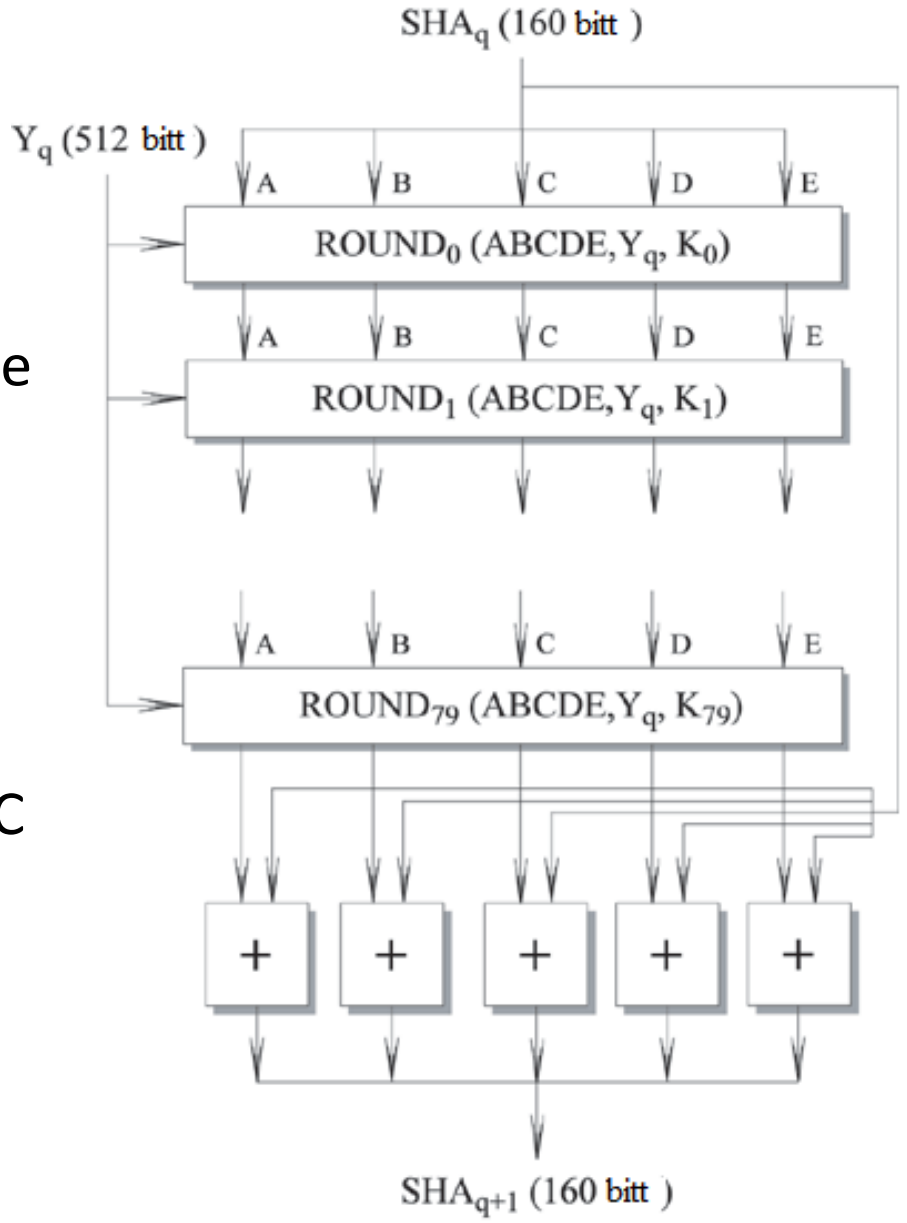
SHA-1 Hash-funktsiooni algoritm.

80 ühesuguse struktuuriga raundi. Igas raundis kasutatakse konstanti k_t.

$0 \leq t \leq 19$ K_t = 5A827999
(täisarvuline osa arvu $[2^{30} \times 2^{1/2}]$)

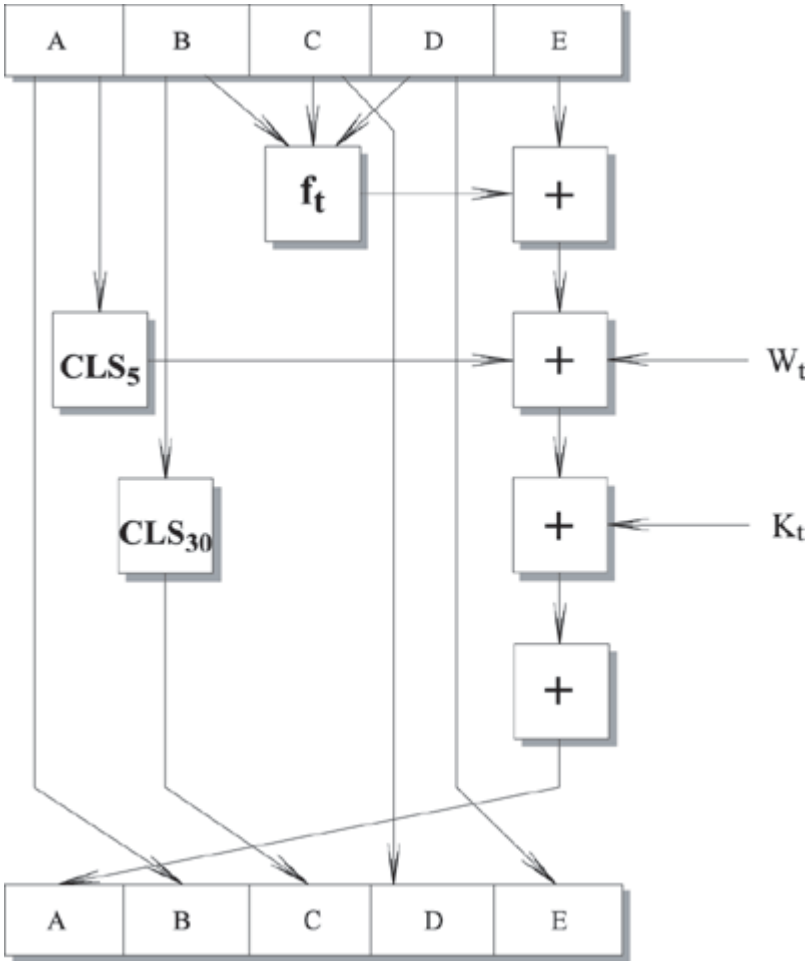
$20 \leq t \leq 39$ K_t = 6ED9EBA1
(täisarvuline osa arvu $[2^{30} \times 3^{1/2}]$) $40 \leq t \leq 59$ K_t = 8F1BBCDC
(täisarvuline osa arvu $[2^{30} \times 5^{1/2}]$)

$60 \leq t \leq 79$ K_t = CA62C1D6
(täisarvuline osa arvu $[2^{30} \times 10^{1/2}]$)



SHA-1 Hash-funktsiooni algoritm. Raund

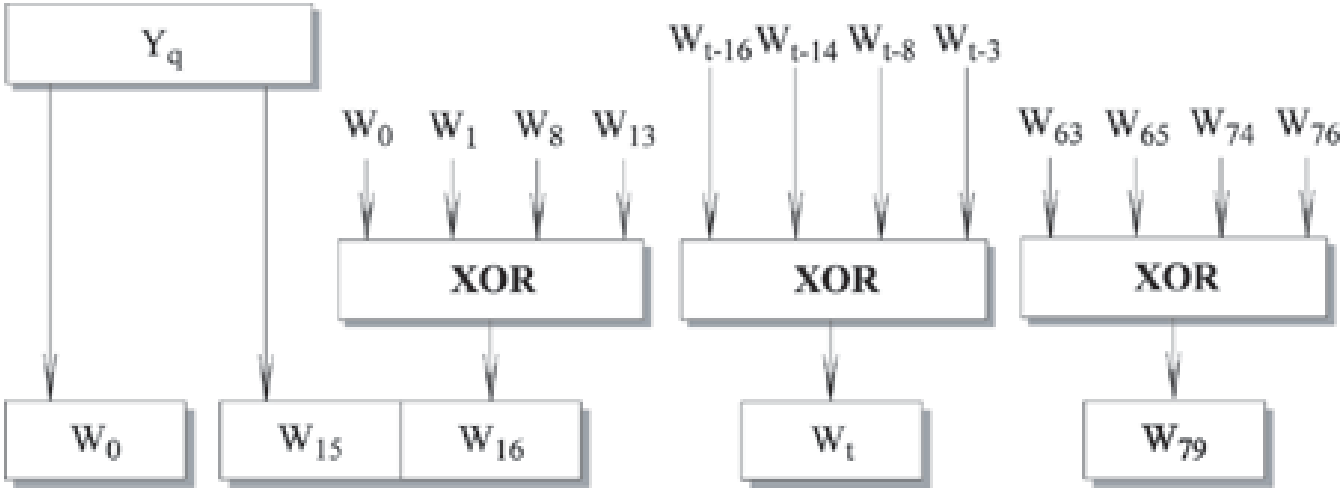
- Iga raundi operatsiooni võib esitada kui A, B, C, D, E (CLS_5 (A) + f_t (B, C, D) + E + W_t + K_t), A, CLS_{30} (B), C, D
- A, B, C, D, E - sisendsõnad.
  t – tsükli number, $0 \leq t \leq 79$.
  f_t – elementaarne funktsioon.
- CLS_s - 32-ndal kohal seisva biti tsükliline nihutamine s biti võrra vasakule.
- W_t - 32-bitine sõna, mis on saadud tsüklile vastavast 512-bitist plokist arvutuste alusel.
- K_t - konstant.
- + - liitmine mooduliga 2^{32}.



SHA-1 Hash-funktsiooni algoritm. Funktsioon f. W_t arvutus.

Raundi number	**ft (B, C, D)**
$(0 \leq t \leq 19)$	$(B \,\&\, C) \vee (\neg B \,\&\, D)$
$(20 \leq t \leq 39)$	$B \oplus C \oplus D$
$(40 \leq t \leq 59)$	$(B \,\&\, C) \vee (B \,\&\, D) \vee (C \,\&\, D)$
$(60 \leq t \leq 79)$	$B \oplus C \oplus D$

W_t arvutus:



Esimest 16 sõna tulevad Y-st.

Järgmised 16 sõna arvutatakse: $W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$

SHA-1 ja MD5 algoritmide võrdlus

	MD5	SHA−1
Hash-koodi pikkus	128 bitti	160 bitti
Töötleva bloki suurus	512 bitti	512 bitti
Iteratsioonide arv	64 (4 raundi, igas 16 iteratsiooni)	80
Elementaarsete loogika funktsioonide arv	4	3
Lisakonstantide arv	64	4

SHA-1 ja MD5 algoritmide võrdlus

- **Turvalisus**. Kuna SHA-1 hash-kood on pikem, siis SHA-1 on vastupidavam otserünnete vastu. On raskem moodustada kaks teksti, millel on võrdne hash-kood ($2^{160} \rightarrow 2^{80}$ SHA-1 puhul ja $2^{128} \rightarrow 2^{64}$ MD5 puhul).
- **Kiirus**. Kuna mõlemad algoritmid kasutavad 32bitist liitmist mod 2^{32}, siis mõlemad vajavad vastavat 32 arhitektuuri. SHA-1 sisaldab rohkem iteratsioone. Võrdse riistvaralise arhidektuuri juures, SHA-1 töötab 25% aeglasem kui MD5.
- **Lihtsus ja kompaktsus.** Mõlemad algoritmid on lihtsad. Seejuures aga SHA-1 rakendab raundides sisendandmete töötlemist ühtemoodi, MD5-s töödeldakse aga ABCD sisendi vastavalt raundile.

Hash-funktsioonid SHA-2

- 2001a NIST võttis kasutusele hash-funktsioone pikema hash-koodiga. Tihti nimetatakse neid konkreetselt: SHA-256, SHA-384 ning SHA-512
- SHA-2el on välja antud patent US patent 6829355

Algoritm	Avateksti pikkus bittides	Plokki pikkus bittides	Sõna pikkus bittides	Hash-koodi pikkus bittides	Sünnipäeva paradoksi piir
SHA-1	$<2^{64}$	512	32	160	80
SHA-256	$<2^{64}$	512	32	256	128
SHA-384	$<2^{128}$	1024	64	384	192
SHA-512	$<2^{128}$	1024	64	512	256

Hash-funktsioonid SHA-2

$Ch(x, y, z) = (x \,\&\, y) \oplus (\neg x \,\&\, z)$

$Maj(x, y, z) = (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z)$

$\Sigma_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$

$\Sigma_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$

$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$

$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$

SHA-256

$Ch(x, y, z) = (x \,\&\, y) \oplus (\neg x \,\&\, z)$

$Maj(x, y, z) = (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z)$

$\Sigma_0^{\{512\}}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x)$

$\Sigma_1^{\{512\}}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)$

$\sigma_0^{\{512\}}(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$

$\sigma_1^{\{512\}}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$

SHA-384 ning
SHA-512

Hash-funktsioonid SHA-2

- Ettevalmistamise etapid on samad, mis SHA-1, (ploki pikkust arvestatakse erinevalt). Tulemuseks on iga avatekst esitatud N plokiga $M^{(1)}$, $M^{(2)}$, … , $M^{(N)}$. Initsialiseeritakse kaheksa sõna (iga sõna vastavalt 32 või 64 bitti): a, b, c, d, e, f, g, h.
- Algoritmi peamiseks osaks on raundid (64 iteratsiooni) iga $M^{(i)}$ jaoks.

$T_1 = h + \Sigma_1^{\{256\}}(e) + Ch(e, f, g) + K_t^{\{256\}} + W_t$

$T_2 = \Sigma_0^{\{256\}}(a) + Maj(a, b, c)$

SHA-256

$h = g$

$g = f$

$f = e$

$e = d + T_1$

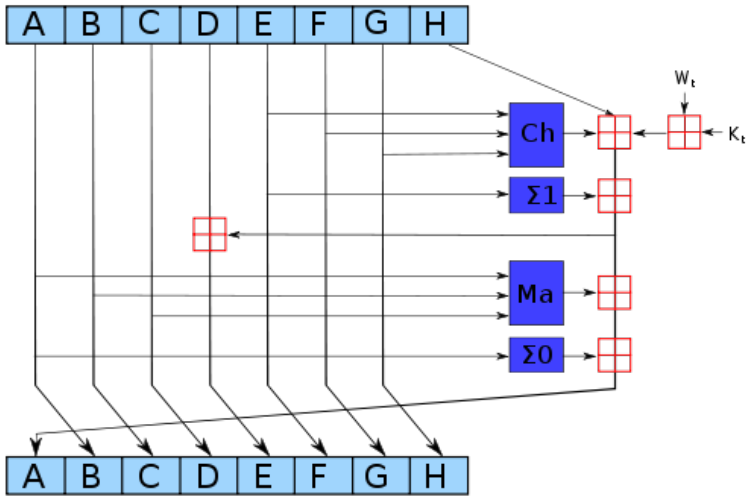
$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

$K_i^{\{256\}}$ on 64 konstanti, mis on esimesed 32 bitti kolmanda astme juurtes, mis on võetud esimesest kuuekümne neljast algarvust.



Hash-funktsioonid SHA-2

- SHA-512 juures on sõna suurus 64 bitti ning 80 iteratsiooni kaheksakümne konstantiga $K_i^{\{512\}}$.

$W_t = M_t^{(i)}$, $0 \leq t \leq 15$

$W_t = \sigma_1^{\{256/512\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256/512\}}(W_{t-15}) + W_{t-16}$,

$16 \leq t \leq 63/79$

$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_7^{(i)} = h + H_7^{(i-1)}$$

Hash-funktsioonide kasutamine

- TLS ja SSL,
- PGP,
- SSH,
- S/MIME,
- IPsec

Autentimine. Autentimise all mõeldakse kinnitust asjaolule, et tekst on saadud just sellelt inimeselt, kellena see isik on end esitlenud.
Kui sõnum on valmis, lisatakse selle lõppu hash-kood. Saatja šifreerib selle koodi enda privaatvõtmega.
Vastuvõtja arvutab “õiget” hash-koodi ning kontrollib tulemust saatja hash-koodiga.